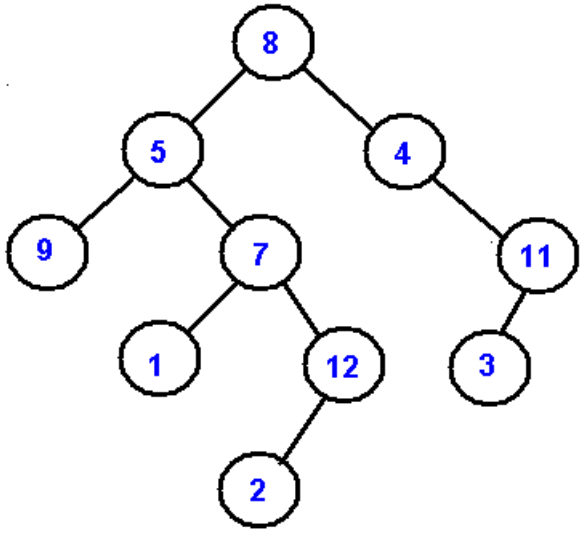
EECS 214 Worksheet 2

For answering the following questions assume that you are given the following tree:



A.Trees

1) Is element 11 a leaf? Who is its parent?

   No, 4

2) How many parents can elements have?

   Answer: 1 (unless it is the root)

3) No node in a binary tree has more than 2 children (true or false)

   True

4) In a post-order traversal, root nodes comes last (true or false)

   True

B.Tree Representations

1) Write a TreeNode class using an array representation to list the children

```
class TreeNode{
TreeNode[] children;
… other data …
}
```

2) Write a TreeNode class for a binary tree

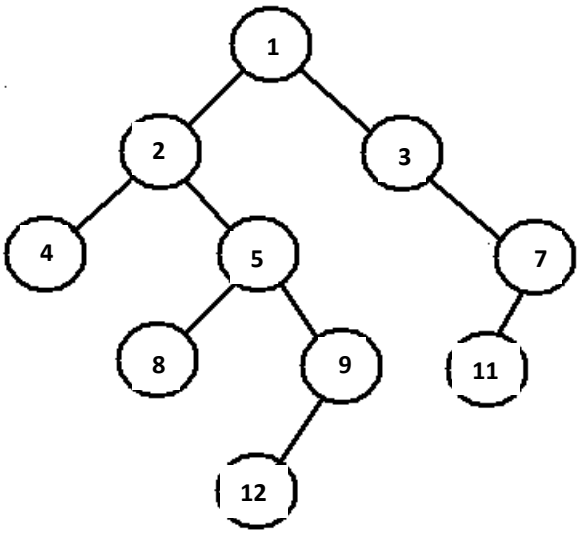
```
class TreeNode{
TreeNode Left;
TreeNode Right;
… other data …
}
```

C. Breadth First Search

1) What is the data-structure you need to keep track of where you've been when you're performing a BFS?

   Answer: Queue

2) Modify the tree above so that the elements would be printed in numerical order if printed by a breadth-first search (draw a tree).



3) Write a BFS function BFSwalk(TreeNode root) to traverse a binary tree

BFSWalk(TreeNode root){

```
  Queue tmp = new Queue();
  tmp.Enqueue(root);

  while (tmp.Count != 0){

      current = tmp.Dequeue();
      Console.Write(current.value);

      tmp.Enqueue(current.Left);
      tmp.Enqueue(current.Right);

  }
}
```

D. Depth First Search

1) How would you modify the BFS function you wrote in C.3) to perform DFS traversal? (explain with words)

   Answer: Use Stack instead of Queue

2) Write three functions to perform pre-order / in-order / post-order traversal of a binary tree (use the execution stack)

   Preorder(TreeNode root){

```
    Console.Write(root.value);

    if (root.Left != null){
            Preorder(root.Left);
    }
    if (root.Right != null){
            Preorder(root.Right);
    }
```

   }

   Inorder(TreeNode root){

```
    if (root.Left != null){
            Inorder(root.Left);
    }
    Console.Write(root.value);

    if (root.Right != null){
            Inorder(root.Right);
    }
```

   }

Postorder(TreeNode root){

```
if (root.Left != null){
        Postorder(root.Left);
}
if (root.Right != null){
        Postorder(root.Right);
}
Console.Write(root.value);
```

}

a. What are the outputs for the tree you are given above?

Preorder: 8,5,9,7,1,12,2,4,11,3

Inorder: 9,5,1,7,2,12,8,4,3,11

Postorder: 9,1,2,12,7,5,3,11,4,8